What is non-static block or Instance Initializer Block?
A class level nameless block that doesn't contain static keyword in its declaration is called non-static block. The non-static block is also called as Instance Initializer Block.

Syntax for creating Instance Initializer block

```
class Example {
        {
                ------------   }  except return and
                ------------   }  throw statement all
                ------------   }  statements are allowed
                ------------   }
        }
}
```

Why non-static block?
It is used for initializing non-static variables and executing some logic common to all constructors only once at the time of object creation.

Q) Who does execute NSB when and where?
NSB is executed automatically by JVM for every object creation in Java stacks area by creating separate stack frame in main thread.

What is the output from the below program?

```
class Example
{
        {
                Sopln("NSB");
        }
        Example(){
                Sopln("No-arg constructor");
        }

        Example(String s){
                Sopln("String-arg constructor");
        }
```

```
public static void main(String[] args){
        System.out.println("main");

        Example e2 = new Example();

        Example e3 = new Example("abc");
    }
}
```

O/P

```
main

NSB
No-arg constructor

NSB
String-arg constructor
```

Q) From this output can you conclude **order of execution** of **constructor and non-static block**?
NSB always executed before constructor logic.

Below program explains initializing non-static variable in instance initialize block common specific to one object common to all constructors

```
import java.util.Scanner;
class Example{
        static Scanner scn = new Scanner(System.in);
        int x;
        {
                System.out.print("\nEnter x value: ");
                x = scn.nextInt();

                System.out.print("NSV is initialized in IIB ");
        }
        Example(){
                System.out.println(" NPC");
        }
        Example(int x){
                System.out.println(" IPC");
        }
        Example(String s){
                System.out.println(" SPC");
        }
        Example(double d){
                System.out.println(" DPC");
        }
}

class Test {
        public static void main(String[] args) {
                Example e1 = new Example();
                Example e2 = new Example(5);
                Example e3 = new Example("a");
                Example e4 = new Example(4.5);
                Example e5 = e4;

                System.out.println();
                System.out.println(e1.x);
                System.out.println(e2.x);
                System.out.println(e3.x);
                System.out.println(e4.x);
        }
}
```

Develop program to count number of objects created from a class

```
class Example{
	private static int count ;
	{
		count++;
	}
	Example(){
		System.out.println("\nobject is created by using NPC");
	}
	Example(int x){
		System.out.println("\nobject is created by using IPC");
	}
	Example(String s){
		System.out.println("\nobject is created by using SPC");
	}
	Example(double d){
		System.out.println("\nobject is created by using DPC");
	}
	public static int getCount(){
		return count;
	}
}

//Test.java
class Test {
	public static void main(String[] args) {
		System.out.println(
			"Number of objects created: "+ Example.getCount());

		System.out.println();
		Example e1 = new Example();
		Example e2 = new Example(5);
		Example e3 = new Example("a");
		Example e4 = new Example("a");
		Example e5 = new Example(5.6);
		System.out.println();

		System.out.println(
			"Number of objects created: "+ Example.getCount());
	}
}
```

Q) In a class, how many instance initializer blocks can be defined?
We can define multiple initializer blocks

Q) What is the order of execution of all Instance Initializer Blocks?
All **Instance Initializer Blocks** are executed by default by JVM in the order they defined from top to bottom before the invoked constructor logic.

What is the output from below program?

```
class Example {

        {
                Sopln("NSB1");
        }

        Example(){
                Sopln("No-arg constructor");
        }

        {
                Sopln("NSB2");
        }

 public static void main(String[] args){
            System.out.println("main");

            Example e2 = new Example();

    }
}
```

O/P

```
main
NSB1
NSB2
No-arg constructor
```

Q) Can we nest instance initializer block?
No, we can’t nest instance initializer block. If we nest instance initializer block, we don’t get compile time error, it is consider as local block.

```
class Example {

        {
                Sopln("Instance Initializer Block");

                {
                        Sopln("Local block");
                }
        }

        Example(){
                Sopln("No-arg constructor");
        }
        public static void main(String[] args){
                System.out.println("main");

                Example e2 = new Example();
        }
}
```